Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection

Michele Orrù, Emmanuela Orsini, Peter Scholl

Actively Secure 1-out-of-N OT Extension with Application to Private Set Intersection *

Michele Orrù[1†], Emmanuela Orsini[2], and Peter Scholl[2]

[1] CNRS, ENS Paris, France
michele.orru@ens.fr
[2] Department of Computer Science, University of Bristol
{Emmanuela.Orsini, Peter.Scholl}@bristol.ac.uk

Abstract. This paper describes a 1-out-of-N oblivious transfer (OT) extension protocol with active security, which achieves very low overhead on top of the passively secure protocol of Kolesnikov and Kumaresan (Crypto 2011). Our protocol obtains active security using a consistency check which requires only simple computation and has a communication overhead that is independent of the total number of OTs to be produced. We prove its security in both the random oracle model and the standard model, assuming a variant of correlation robustness. We describe an implementation, which demonstrates our protocol only costs around 5–30% more than the passively secure protocol.
Random 1-out-of-N OT is a key building block in recent, very efficient, passively secure private set intersection (PSI) protocols. Our random OT extension protocol has the interesting feature that it even works when N is exponentially large in the security parameter, provided that the sender only needs to obtain polynomially many outputs. We show that this can be directly applied to improve the performance of PSI, allowing the core private equality test and private set inclusion subprotocols to be carried out using just a single OT each. This leads to a reduction in communication of up to 3 times for the main component of PSI.

Keywords: Oblivious transfer; private set intersection; multi-party computation

*This is the full version of a paper published at CT-RSA 2017. The final publication is available at http://link.springer.com.

†Work done while visiting University of Bristol

Table of Contents

1 Introduction

Oblivious transfer (OT) is a fundamental primitive in cryptography, first introduced by Rabin [Rab81] and now employed in a variety of protocols, ranging from contract signing [EGL85] to special-purpose tasks such as private set intersection [PSZ14]. It plays a decisive role in protocols for secure two-party and multi-party computation, including those based on Yao's garbled circuits [Yao82] and secret-sharing [NNOB12,LOS14,KOS16]. The most commonly studied form of oblivious transfer is 1-out-of-2 OT, where a sender has two messages (x_0, x_1) as input, and a receiver chooses a bit b; the goal of the protocol is for the receiver to learn x_b, but no information on x_{1-b}, whilst the sender learns nothing about b. This can be generalized to 1-out-of-N OT and k-out-of-N OT, in which the receiver learns k of the sender's N messages.

Unfortunately, due to a result of Impagliazzo and Rudich [IR89], oblivious transfer is highly unlikely to be possible without the use of public-key cryptography; consequently, even the most efficient oblivious transfer constructions [PVW08,CO15] come with a relatively high cost.

OT Extensions. In 1996, Beaver [Bea96] first showed that it is possible to extend OT starting with a small number (say, security parameter κ) of "base" OTs, to create $\mathsf{poly}(\kappa)$ additional OTs using only symmetric primitives, with computational security κ. This construction is very impractical as it requires the evaluation of pseudorandom generators within Yao's garbled circuits.

Later, in 2003, Ishai et al. [IKNP03] proposed a protocol for extending oblivious transfers: the passively secure version of this protocol (hereafter IKNP) only requires black-box use of a correlation robust hash function, and is very efficient. Concretely, an optimized version of IKNP for OT on random strings (described in [ALSZ13,KK13]) requires sending κ bits and computing three hash function evaluations per OT, after a one-time cost of κ base OTs, for computational security κ. With a carefully optimized implementation, the dominant cost of this is communication [ALSZ16].

Kolesnikov and Kumaresan [KK13] showed how to modify the IKNP protocol using Walsh-Hadamard error-correcting codes and obtain a passively secure protocol for 1-out-of-N OT on random strings. The cost is only a small constant factor more than the 1-out-of-2 IKNP for values of N up 256.

Several recent works have proposed increasingly efficient protocols for 1-out-of-2 OT extension with active security [NNOB12,ALSZ15,KOS15]. The latter work of Keller et al. [KOS15], which is proven secure in the random oracle model, brings the cost of actively secure 1-out-of-2 OT to essentially the same as the passive IKNP protocol by adding a simple consistency check.

1.1 Contributions

Actively Secure 1-out-of-N OT Extension. Our main contribution is a practical, actively secure 1-out-of-N OT extension protocol with very low overhead on top of the passively secure protocol of Kolesnikov and Kumaresan [KK13]. For the case of random OT, where the sender's strings are sampled at random, our protocol (proven secure in the random oracle model) improves upon [KK13] by allowing for much larger values of N with a suitable choice of binary linear code. Our protocol even works when N is *exponential in the security parameter*, provided that the sender is only required to learn polynomially many output strings. The protocol requires only κ base OTs, and the extension phase has an amortized communication cost of $O(\kappa)$ bits per random OT.

At a high level, our protocol starts with the passively secure [KK13] protocol and adds a simple consistency check to obtain active security (similar to [KOS15] for 1-out-of-2 OT). However, there

are several technical challenges to solve on the way. In [KOS15], a check is used to verify that pairs of strings are of the form $(\mathbf{x}_i, \mathbf{x}_i + \mathbf{b})$ for a fixed correlation $\mathbf{b}$ (with addition modulo 2), when the receiver only knows one string from each pair. In the [KK13] protocol, however, we must ensure that strings are of the form $\mathbf{x}_i + \mathbf{b} \odot \mathcal{C}(m_i)$, where $\mathcal{C}$ encodes a message m_i using an error-correcting code and $\odot$ denotes the component-wise product of bit vectors. The check of [KOS15] cannot be applied to this situation. We overcome this by adapting a check used previously in additively homomorphic UC commitments [FJNT16], which requires that $\mathcal{C}$ is a *linear* code with sufficiently large minimum distance. [1] The number of codewords in the binary linear code determines N in the 1-out-of-N OT, which gives a range of choices of N depending on the choice of code.

To be able to handle exponentially large N, it may seem that we just need to choose a suitable binary linear code of the right length. However, we need to take care that the security reduction does not contain any loss in security that scales with N: the reduction in [KK13] incurs a loss in $O(N^2)$, which would give a meaningless security result in this case. To ensure this, we modify the 1-out-of-N random OT functionality so that the sender can only obtain $N' = \mathsf{poly}(\kappa)$ of the output messages, and show that the loss in the resulting reduction is in $O(N')$.

Security in the Standard Model. For random OT extension, it is not known how to prove security without using a programmable random oracle as in [ALSZ13] and [KOS15]. However, for the case of non-random 1-out-of-N OT, we prove our protocol secure in the standard model, assuming a hash function that satisfies a variant of correlation robustness on high min-entropy secrets. This is a similar assumption to the protocol in [ALSZ15], but more general as we require the assumption to hold for a range of different parameters. This gives the first actively secure OT extension protocol needing only κ base OTs for security parameter κ and is proven secure without random oracles, even in the 1-out-of-2 case. [2]

Faster Private Set Intersection. We show that random 1-out-of-N OT with an exponentially large N can be directly applied to improve the efficiency of the previous fastest (semi-honest) private set intersection protocols. OT-based PSI protocols [PSZ14,PSSZ15] use random 1-out-of-N OT as a building block for a private equality test protocol, where two parties learn whether their inputs are equal (and nothing more). In that protocol, one random OT is used to perform an equality test on $\log N$-bit inputs. Since the random OT protocol of [KK13] only works for values of N up to 256 (due to the use of small Walsh-Hadamard codes) several OTs are XORed together to construct a protocol for comparing large (e.g. up to 128 bit) messages. Using our protocol with $N = 2^k$ gives a very simple private equality test on k-bit messages, for any $k = \mathsf{poly}(\kappa)$, using just a single 1-out-of-N random OT. This can be generalized to perform *private set inclusion* — where one party holds a single value and another party a set of m values — at the cost of one random OT and sending $m \cdot s$ bits, where s is the statistical security parameter. This results in a reduction in communication of around 2–5 times (depending on the bit-length of the input) for this component of the semi-honest PSI protocol in [PSSZ15].

[1] We observe an interesting connection between our protocol and additively homomorphic UC commitment schemes [FJNT16,CDD+16]: our protocol essentially runs a homomorphic commitment protocol and hashes the resulting commitments to obtain random OTs. However, this mechanism seems very specific to the workings of these commitment schemes and appears unlikely to lead to a generic transformation.

[2] Note that our security reduction requires fixing the adversary's random coins, so is non-uniform. Obtaining a uniform reduction seems to need at least $\kappa + s$ base OTs, for statistical security parameter s.

Implementation. We have implemented and benchmarked our 1-out-of-N random OT extension protocol and compared its performance with the passive protocol of [KK13]. Although our implementation is not heavily optimized (it occupies around 800 lines of C in all), we show that the overhead of our consistency check for achieving active security is very low: the actively secure protocol takes only around 20% more time than the passive version, depending on parameters.

Towards Efficient Actively Secure PSI. Currently, the most efficient PSI protocols are the OT-based ones mentioned above, but these are only secure against a passive adversary. Since 1-out-of-N random OT is a key component in these protocols, our work can be seen as a step towards constructing more efficient PSI with active security. Actively secure PSI was recently studied by Lambæk [Lam16], who showed the protocol of [PSSZ15] can be modified to provide active security for one party, assuming the underlying OT protocol is actively secure; our protocol therefore provides an instantiation of this proposal.

Recent Work and Open Problems. In a very recent, independent work, Kolesnikov et al. [KKRT16] describe a batched oblivious PRF evaluation protocol with application to private set intersection. Although their protocol is phrased in the language of oblivious PRFs rather than 1-out-of-N OT, it is very similar to ours, only with passive security. Instead of using a traditional error-correcting code, they show that a random oracle has the necessary properties for passive security. In contrast, our protocol requires the linearity and erasure decoding properties of the binary code to achieve active security. They describe the same application to improved performance of PSI (with slightly better parameters than ours due to use of a random oracle) and give a thorough efficiency evaluation and implementation of the resulting PSI protocol. We note that it is still an interesting open problem to obtain a fully actively secure variant of the PSI protocol in [PSSZ15] with low overhead.

Regarding OT extension in general, there are still some interesting unsolved problems. Our 1-out-of-N OT extension cannot be used directly to improve performance of 1-out-of-2 OT on short secrets (as was done for the passive case in [KK13]), since the standard reduction from 1-out-of-N to 1-out-of-2 OT [NP99] is only passively secure. Therefore, it is still an open problem to construct a practical 1-out-of-2 OT extension on short strings with communication sublinear in the security parameter. Also, the case of constructing k-out-of-N OT with active security using OT extensions is still open; there is an elegant passively secure protocol [SSR08], but it seems difficult to make this actively secure.

2 Preliminaries

Notation. We denote by κ and s the computational, resp. statistical, security parameters. We use bold lower case letters for vectors. Given a matrix A, we let $\mathbf{a}_i$ denote the i-th row of A, and $\mathbf{a}^j$ denote the j-th column of A. When referring to a vector $\mathbf{v} \in \mathbb{F}^n$, we write $\mathbf{v}[i]$, with $1 \leq i \leq n$, to mean the i-th component of $\mathbf{v}$. We identify bit strings as vectors over the finite field $\mathbb{F}_2$, and use "$+$" and "$\cdot$" to mean addition and multiplication in this field. We use the notation $\mathbf{a} \odot \mathbf{b}$ to denote the component-wise product of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$. Given an integer N, we denote by $[N]$ the set of integers $\{1, \ldots, N\}$.

Error-correcting Codes. Our protocol uses an $[n_{\mathcal{C}}, k_{\mathcal{C}}, d_{\mathcal{C}}]$ binary linear code $\mathcal{C}$, where $n_{\mathcal{C}}$ is the length, $k_{\mathcal{C}}$ the dimension and $d_{\mathcal{C}}$ the distance of $\mathcal{C}$. So, $\mathcal{C} : \mathbb{F}_2^{k_{\mathcal{C}}} \to \mathbb{F}_2^{n_{\mathcal{C}}}$ is a linear map such that for every pair of messages $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{F}_2^{k_{\mathcal{C}}}$, the Hamming weight of the sum of the encodings of the messages satisfies $\mathsf{wt}_{\mathsf{H}}(\mathcal{C}(\mathbf{m}_1) + \mathcal{C}(\mathbf{m}_2)) \geq d_{\mathcal{C}}$.

Oblivious Transfer Functionalities. We now recall some definitions of oblivious transfer. Following Even et al. [EGL85], 1-out-of-2 OT is a two-party protocol between a sender $\mathrm{P_S}$, who inputs two messages v_0, v_1, and a receiver $\mathrm{P_R}$ who inputs a choice bit c and learns as output v_c and nothing about v_{1-c}, in such a way that $\mathrm{P_S}$ remains oblivious as what message was received by $\mathrm{P_R}$. Formally, the general case of 1-out-of-N OT on κ-bit strings is defined as the functionality:

$$\mathcal{F}_{N\text{-}\mathsf{OT}}((\mathbf{v}_0, \ldots, \mathbf{v}_{N-1}), c) = (\perp, \mathbf{v}_c),$$

where $\mathbf{x}_i \in \{0,1\}^\kappa$ are the sender's inputs and $c \in \{0, \ldots, N-1\}$ is the receiver's input. We denote by $\mathcal{F}^{\kappa,m}_{N\text{-}\mathsf{OT}}$ the functionality that runs $\mathcal{F}_{N\text{-}\mathsf{OT}}$ m times on messages in $\{0,1\}^\kappa$. For example, in $\mathcal{F}^{\kappa,m}_{2\text{-}\mathsf{OT}}$, P_S inputs $(\mathbf{v}_{i,0}, \mathbf{v}_{i,1})$ and P_R inputs c_i for $i \in [m]$, and P_R receives the output $\mathbf{v}_{i,c_i}$.

Another important variant is the random OT functionality $\mathcal{F}^{\kappa,m}_{N\text{-}\mathsf{ROT}}$, in which the sender provides no input, but receives random messages $(\mathbf{v}_0, \ldots, \mathbf{v}_{N-1})$ from the functionality as output.

2.1 Passively Secure OT Extension: the KK Protocol

We now recall the passively secure KK protocol for 1-out-of-N OT extension described in [KK13], which is a generalized version of the IKNP protocol for 1-out-of-2 OT [IKNP03].

Suppose the two parties wish to perform m sets of 1-out-of-N random OTs, where N is a power of two. There is a sender $\mathrm{P_S}$ with no input, and a receiver $\mathrm{P_R}$, who inputs the choices $w_1, \ldots, w_m \in \{0, \ldots, N-1\}$, which are represented as vectors $\mathbf{w}_i \in \mathbb{F}_2^{\log N}$. The two parties begin by performing $n_{\mathcal{C}}$ base 1-out-of-2 OTs on random inputs, with the roles of sender and receiver reversed. So, P_R obtains $n_{\mathcal{C}}$ pairs of random strings $(\mathbf{r}_0^j, \mathbf{r}_1^j)$ of length κ and P_S obtains $(b_j, \mathbf{r}_{b_j}^j)$, where $b_j \xleftarrow{\$} \{0,1\}$, for $j \in [n_{\mathcal{C}}]$.

Next, both parties locally extend their base OT outputs to length m using a pseudorandom generator, where m is the final number of OTs desired. This results in κ sets of 1-out-of-2 OTs on m-bit strings, which we represent as matrices $T_0, T_1 \in \mathbb{F}_2^{m \times n_{\mathcal{C}}}$, held by $\mathrm{P_R}$, whilst $\mathrm{P_S}$ holds the vector $\mathbf{b} = (b_1, \ldots, b_{n_{\mathcal{C}}}) \in \mathbb{F}_2^{n_{\mathcal{C}}}$ and the matrix

$$T_{\mathbf{b}} := \left(\mathbf{t}_{b_1}^1 \ldots \mathbf{t}_{b_{n_{\mathcal{C}}}}^{n_{\mathcal{C}}}\right) \in \mathbb{F}_2^{m \times n_{\mathcal{C}}},$$

where $\mathbf{t}_0^j, \mathbf{t}_1^j$ are the columns of T_0, T_1, for $j \in [n_{\mathcal{C}}]$.

At this point $\mathrm{P_R}$ constructs a matrix $C \in \mathbb{F}_2^{m \times n_{\mathcal{C}}}$, where each row $\mathbf{c}_i$ is the encoding $\mathcal{C}(\mathbf{w}_i)$ of the input $\mathbf{w}_i \in \mathbb{F}_2^{k_{\mathcal{C}}}$, where $\mathcal{C}$ is a binary code of length $n_{\mathcal{C}}$, dimension $k_{\mathcal{C}} = \log_2 N$ and minimum distance $d_{\mathcal{C}} \geq \kappa$. Then $\mathrm{P_R}$ sends to $\mathrm{P_S}$ the matrix

$$U = T_0 + T_1 + C.$$

Note that for each column of U, all information on the receiver's encoded input is masked by the value $\mathbf{t}_{1-b_j}^j$, which is unknown to $\mathrm{P_S}$.

After this step $\mathrm{P_S}$ defines an $m \times n_{\mathcal{C}}$ matrix Q with columns $\mathbf{q}^j = b_j \cdot \mathbf{u}^j + \mathbf{t}_{b_j}^j = b_j \cdot \mathbf{c}^j + \mathbf{t}_0^j$ (where $\mathbf{c}^j$ are the columns of C). Notice that the rows of Q are given by

$$\mathbf{q}_i = \mathbf{c}_i \odot \mathbf{b} + \mathbf{t}_i,$$

where $\mathbf{t}_i$ are the rows of T_0. Here, $\mathrm{P_R}$ holds $\mathbf{t}_i$ and $\mathrm{P_S}$ holds $(\mathbf{q}_i, \mathbf{b})$, for $i \in [m]$. The key observation to turn these values into OTs is that for each of the possible receiver choices $\mathbf{w} \in \mathbb{F}_2^{k_{\mathcal{C}}}$, $\mathrm{P_S}$ can

compute the value $\mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b}$. If $\mathbf{w} = \mathbf{w}_i$ then this is equal to $\mathbf{t}_i$ so is known to $\mathrm{P_R}$. Otherwise, for any $\mathbf{w} \neq \mathbf{w}_i$, $\mathrm{P_R}$ must guess κ bits of $\mathrm{P_S}$'s secret $\mathbf{b}$ to be able to compute $\mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b}$, since the minimum distance of $\mathcal{C}$ guarantees that $\mathcal{C}(\mathbf{w})$ and $\mathcal{C}(\mathbf{w}_i)$ are at least Hamming distance κ apart.

Therefore, the parties can convert these values to random 1-out-of-N OTs by simply hashing their outputs with a random oracle, H. $\mathrm{P_S}$ outputs the values $\mathbf{v}_{w,i} = \mathsf{H}(i, \mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})$, for all $\mathbf{w} \in \mathbb{F}_2^{k_C}$, and $\mathrm{P_R}$ outputs $\mathbf{v}_{w_i,i} = \mathsf{H}(i, \mathbf{t}_i)$.

Instantiating the Code. As noticed in [KK13], if we instantiate the binary linear code $\mathcal{C}$ with the $[\kappa, 1, \kappa]$ binary repetition code, we obtain the 1-out-of-2 IKNP protocol [IKNP03]. In this case, each row of the matrix C constructed by the receiver is either 0^κ or 1^κ, depending on the receiver's choice bits. If instead $\mathcal{C}$ is chosen to be a Walsh-Hadamard code as in [KK13], then the result is a 1-out-of-2^{k_C} OT. This needs a code length of $N = 2^{k_C}$ with security parameter $N/2$; this turns out to be more efficient than constructing 1-out-of-N OT from 1-out-of-2 OT for values of $N \leq 256$ with 128-bit security.

Security. The KK protocol (and hence IKNP) is actively secure against a corrupt sender, since after the base OTs, there is no opportunity for $\mathrm{P_S}$ to cheat. However, it only provides passive security against a corrupt receiver, since $\mathrm{P_R}$ may incorrect compute the encodings of their input in the matrix U. It was explained in [IKNP03,ALSZ15] that if $\mathrm{P_R}$ cheats in this way, and also learns (via a side-channel, for instance) the sender's outputs in just κ of the random OTs then $\mathrm{P_R}$ can compute the sender's secret $\mathbf{b}$, and thus learn all of the sender's outputs in every remaining OT.

3 Actively Secure Random 1-out-of-N OT Extension

In this section we present our actively secure OT extension protocol in the random oracle model. Since we want to construct 1-out-of-N random OT when N may be exponential in the security parameter, our protocol implements a modified random OT functionality $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ (Fig. 1), which allows the sender to query the functionality to obtain their random OT outputs one at a time, so that all N need not be produced.

Functionality $\mathcal{F}^{\kappa,m}_{N\text{-}\mathsf{ROT}+}$

Upon receiving (ROT) from $\mathrm{P_S}$ and $(\mathsf{ROT}, (w_1, \ldots, w_m))$, where $\forall i \in [m], 0 \leq w_i < N$, from $\mathrm{P_R}$, do the following:

- Sample $\mathbf{v}_{j,i} \xleftarrow{\$} \mathbb{F}_2^\kappa$ for each $j \in \{0, \ldots, N-1\}$ and $i \in [m]$.
- Send the outputs $\mathbf{v}_{w_i,i}$ to $\mathrm{P_R}$ for each $i \in [m]$
- Upon receiving $(\mathsf{SenderOutput}, j, i)$ from $\mathrm{P_S}$, where $i \in [m]$ and $j \in \{0, \ldots, N-1\}$, send $\mathbf{v}_{j,i}$ to $\mathrm{P_S}$.

Fig. 1: Ideal functionality $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ for m 1-out-of-$(\leq N)$ random OTs on κ-bit strings between a sender $\mathrm{P_S}$ and receiver $\mathrm{P_R}$

The high-level idea of our protocol (Fig. 2) is that, to deal with a malicious receiver in the KK protocol, we add a consistency check that ensures $\mathrm{P_R}$ inputs codewords as rows of the matrix C when sending the matrix U in step 3. If the check passes then the protocol carries on and the correlated OTs are hashed to obtain random OTs. Otherwise, the protocol aborts.

Protocol N-$\mathsf{ROT}^{\kappa,m}$

COMMON INPUT: κ and s are the computational and statistical security parameters, respectively; $\mathcal{C}$ is an $[n_{\mathcal{C}}, k_{\mathcal{C}}, d_{\mathcal{C}}]$ binary linear code such that $k_{\mathcal{C}} = \log_2 N$ and $d_{\mathcal{C}} \geq \kappa$.
INPUT OF $\mathrm{P_R}$: m selection integers $(w_1, \ldots, w_m)$, each in $\{0, \ldots, N-1\}$, encoded as bit strings $\mathbf{w}_1, \ldots, \mathbf{w}_m \in \mathbb{F}_2^{k_{\mathcal{C}}}$.
INPUT OF $\mathrm{P_S}$: m subsets $S_i \subseteq \mathbb{F}_2^{k_{\mathcal{C}}}$, with $|S_i| = \mathsf{poly}(\kappa)$, for $i \in [m]$.
REQUIRE: $\mathsf{H} : [m] \times \mathbb{F}_2^{n_{\mathcal{C}}} \to \mathbb{F}_2^{\kappa}$ is a random oracle and $\mathsf{PRG} : \{0,1\}^{\kappa} \to \{0,1\}^{m'}$, $m' = m + s$, is a pseudorandom generator.

Init: Both parties call $\mathcal{F}_{\text{2-OT}}^{\kappa, n_{\mathcal{C}}}$, with $\mathrm{P_R}$ playing the role of the sender and $\mathrm{P_S}$ playing the role of the receiver, inputing $n_{\mathcal{C}}$ random bits. $\mathrm{P_S}$ receives $\{(b_j, \mathbf{r}_{b_j}^j)\}_{j \in [n_{\mathcal{C}}]} \in \mathbb{F}_2 \times \mathbb{F}_2^{\kappa}$ and $\mathrm{P_R}$ receives $\{(\mathbf{r}_0^j, \mathbf{r}_1^j)\}_{j \in [n_{\mathcal{C}}]}$.

Extend: Let $m' = m + s$.

1. $\mathrm{P_R}$ constructs matrices $T_0, T_1 \in \mathbb{F}_2^{m' \times n_{\mathcal{C}}}$ from seeds $\{(\mathbf{r}_0^j, \mathbf{r}_1^j)\}$ so that the respective columns are:

$$\mathbf{t}_0^j = \mathsf{PRG}(\mathbf{r}_0^j) \in \mathbb{F}_2^{m'}, \qquad \mathbf{t}_1^j = \mathsf{PRG}(\mathbf{r}_1^j) \in \mathbb{F}_2^{m'}, \quad j \in [n_{\mathcal{C}}].$$

   In the same way $\mathrm{P_S}$ produces $\mathbf{t}_{b_j}^j$, for each $j \in [n_{\mathcal{C}}]$. Summarizing, $\mathrm{P_R}$ holds $\{(\mathbf{t}_0^j, \mathbf{t}_1^j)\}_{j \in [n_{\mathcal{C}}]}$ and $\mathrm{P_S}$ holds $\{\mathbf{t}_{b_j}^j\}_{j \in [n_{\mathcal{C}}]}$

2. $\mathrm{P_R}$ samples random $\mathbf{w}_{m+\ell} \xleftarrow{\$} \mathbb{F}_2^{k_{\mathcal{C}}}$, for $\ell \in [s]$, and then constructs a matrix $C \in \mathbb{F}_2^{m' \times n_{\mathcal{C}}}$ such that each row $\mathbf{c}_i$ is the codeword $\mathcal{C}(\mathbf{w}_i)$. Then, $\mathrm{P_R}$ sends to $\mathrm{P_S}$ the values

$$\mathbf{u}^j = \mathbf{t}_0^j + \mathbf{t}_1^j + \mathbf{c}^j, \quad j \in [n_{\mathcal{C}}], \tag{1}$$

   where $\mathbf{c}^j$ is the j-th column of C.

3. $\mathrm{P_S}$ receives $\mathbf{u}^j \in \mathbb{F}_2^{m'}$ and computes

$$\mathbf{q}^j = b_j \cdot \mathbf{u}^j + \mathbf{t}_{b_j}^j = b_j \cdot \mathbf{c}^j + \mathbf{t}_0^j, \tag{2}$$

   that form the columns of an $(m' \times n_{\mathcal{C}})$ matrix Q. Denoting the rows of T_0, Q by $\mathbf{t}_i, \mathbf{q}_i$, $\mathrm{P_R}$ now holds $\mathbf{c}_i, \mathbf{t}_i$ and $\mathrm{P_S}$ holds $\mathbf{b}, \mathbf{q}_i$ so that

$$\mathbf{q}_i = \mathbf{c}_i \odot \mathbf{b} + \mathbf{t}_i.$$

4. *Consistency check*:
   - $\mathrm{P_S}$ samples s random strings $\{(x_1^{(\ell)}, \ldots, x_m^{(\ell)}) \in \mathbb{F}_2^m\}_{\ell \in [s]}$ and sends them to $\mathrm{P_R}$.
   - $\mathrm{P_R}$ computes and sends, for $\ell \in [s]$:

$$\mathbf{t}^{(\ell)} = \sum_{i \in [m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell}, \qquad \mathbf{w}^{(\ell)} = \sum_{i \in [m]} \mathbf{w}_i \cdot x_i^{(\ell)} + \mathbf{w}_{m+\ell}$$

   - $\mathrm{P_S}$ computes $\mathbf{q}^{(\ell)} = \sum_{i \in [m]} \mathbf{q}_i \cdot x_i^{(\ell)} + \mathbf{q}_{m+\ell}$, and checks that:

$$\mathbf{t}^{(\ell)} + \mathbf{q}^{(\ell)} = \mathcal{C}(\mathbf{w}^{(\ell)}) \odot \mathbf{b}, \quad \forall \ell \in [s]. \tag{3}$$

   If the check fails, $\mathrm{P_S}$ sends Abort, otherwise continue.

Output: $\mathrm{P_S}$ sets $\forall i \in [m]$ and $\mathbf{w} \in S_i$: $\mathbf{v}_{w,i} = \mathsf{H}(i, \mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})$ and $\mathrm{P_R}$ sets $\forall i \in [m]$: $\mathbf{v}_{w_i,i} = \mathsf{H}(i, \mathbf{t}_i)$.

Fig. 2: An actively secure protocol for $\mathcal{F}_{N\text{-}\mathsf{ROT}+}^{\kappa,m}$, extending $\mathcal{F}_{\text{2-OT}}^{\kappa,n_{\mathcal{C}}}$.

We begin by running the KK protocol to create $m' = m + s$ OTs, where s is the statistical security parameter, in steps 1–3. Recall from the previous section that after running KK, the receiver holds vectors $\mathbf{t}_i, \mathbf{c}_i$ and the sender holds $\mathbf{b} = (b_1, \ldots, b_\kappa) \in \mathbb{F}_2^{\kappa}$ and $\mathbf{q}_i$ such that

$$\mathbf{q}_i = \mathbf{t}_i + \mathbf{c}_i \odot \mathbf{b}, \quad i \in [m'].$$

If $\mathrm{P_R}$ is honest then $\mathbf{c}_i = \mathcal{C}(\mathbf{w}_i)$, where $\mathbf{w}_i \in \mathbb{F}_2^{k_{\mathcal{C}}}$ represents the i-th choice of $\mathrm{P_R}$.

To verify this, $\mathrm{P_S}$ samples s random m-bit strings $(x_1^{(\ell)}, \ldots, x_m^{(\ell)}) \in \mathbb{F}_2^{\ell}$, for $\ell \in [s]$, and sends these to $\mathrm{P_R}$, who then computes for each ℓ the linear combinations

$$\mathbf{t}^{(\ell)} = \sum_{i\in[m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell}, \quad \mathbf{w}^{(\ell)} = \sum_{i\in[m]} \mathbf{w}_i \cdot x_i^{(\ell)} + \mathbf{w}_{m+\ell}$$

and sends these to $\mathrm{P_S}$. Note that adding the random, dummy OT values $\mathbf{t}_{m+\ell}$ and $\mathbf{w}_{m+\ell}$ ensures that the inputs of $\mathrm{P_R}$ are perfectly masked in this step.

Next, $\mathrm{P_S}$ computes $\mathbf{q}^{(\ell)} = \sum_{i\in[m]} \mathbf{q}_i \cdot x_i^{(\ell)} + \mathbf{q}_{m+\ell}$ and checks that $\mathbf{q}^{(\ell)} = \mathbf{t}^{(\ell)} + \mathcal{C}(\mathbf{w}^{(\ell)}) \odot \mathbf{b}$ for each $\ell \in [s]$. If the check goes through, both parties hash their outputs to obtain the random OTs.

The intuition behind security is that if not all the $\mathrm{P_R}$'s inputs $\mathbf{c}_i$ are codewords then to pass the check, the errors must 'cancel out' when taking the random linear combinations. However, the $x_i^{(\ell)}$ values used in the consistency check are unknown when $\mathrm{P_R}$ chooses $\mathbf{c}_i$ so this can only happen with negligible probability; since each $x_i^{(\ell)} \in \{0, 1\}$, there is a $1/2$ probability that $\mathbf{c}_i$ is not included in the linear combination, so s sets of checks are needed to ensure a negligible cheating probability.

Compared with the consistency check of [KOS15] (for the 1-out-of-2 case), our check is simpler as we only require XOR operations instead of multiplications in the finite field $\mathbb{F}_{2^\kappa}$. However, being over $\mathbb{F}_2$ means that we must repeat the check s times, whereas [KOS15] only needs one check; in our case, working in $\mathbb{F}_{2^\kappa}$ would not allow the linear encoding relation to be verified, which is why we use $\mathbb{F}_2$.

We observe that our protocol, minus the final hashing step, is essentially the same as the additively homomorphic commitment protocol from [FJNT16] (which inspired our consistency check). Although our security proof requires quite some extra work to implement OT instead of commitments, it is interesting to see how the same construction can lead to two very different applications with just a small modification. More recently, another scheme [CDD+16] improved upon [FJNT16] by using a linear-time computable consistency check based on a special class of universal hash functions, and constructing a linear-time encodable error-correcting code. These changes can also be applied to our protocol, but it is not clear how efficient these would be in practice, and since we aim for practical (rather than asymptotic) efficiency we do not present this here.

Theorem 1. *Assuming that* H *is a random oracle and* PRG *a pseudo-random generator, the protocol* N-$\mathsf{ROT}^{\kappa,m}$ *in Fig. 2 securely implements* $\mathcal{F}_{N\text{-}\mathsf{ROT}+}^{\kappa,m}$ *(Fig. 1) in the* $\mathcal{F}_{2\text{-}\mathsf{OT}}$*-hybrid model with computational security parameter* κ *and statistical security parameter* s *against a static malicious adversary.*

The case of a corrupt sender is straightforward and reduces to the security of PRG, similar to previous works [ALSZ16,KOS15]. For a corrupt receiver, the first main challenge is for the simulator to extract the receiver's inputs, $\mathbf{w}_i$, to send to the functionality $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$. This is done by using the values sent during the check to identify a set of positions where the receiver has attempted to 'guess' some bits of the sender's secret, $\mathbf{b}$. Removing these positions from the $\mathbf{c}_i$ values used by $\mathrm{P_R}$ leaves behind an incomplete codeword, which can be erasure decoded to recover the message.

After decoding the inputs, the simulator must then respond to random oracle queries made by the environment. We do this in an *optimistic* manner, meaning, we do not abort if conflicting queries are made, but answer at random in that case; the environment may not always notice this inaccurate behaviour if the sender did not learn all N outputs from the OTs. This allows us to obtain a security bound that depends on N', the *maximum* number of outputs learnt by $\mathrm{P_S}$ in any OT, rather than N, which may be exponential in κ.

Proof. First of all we show that the consistency check will always pass in case of honest parties. Upon receiving $\mathbf{t}^{(\ell)}$ and $\mathbf{w}^{(\ell)}$, $\forall \ell \in [s]$, from $\mathrm{P_R}$, the sender $\mathrm{P_S}$ computes

$$\begin{aligned}
\mathbf{t}^{(\ell)} + \mathbf{q}^{(\ell)} =& \Big(\sum_{i\in[m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell}\Big) + \Big(\sum_{i\in[m]} \mathbf{q}_i \cdot x_i^{(\ell)} + \mathbf{q}_{m+\ell}\Big) \\
=& \sum_{i\in[m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell} + \sum_{i\in[m]} \big(\mathbf{c}_i \odot \mathbf{b} + \mathbf{t}_i\big) \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell} \odot \mathbf{b} + \mathbf{t}_{m+\ell} \\
=& \sum_{i\in[m]} (\mathbf{c}_i \odot \mathbf{b}) \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell} \odot \mathbf{b} = \sum_{i\in[m]} (\mathcal{C}(\mathbf{w}_i) \odot \mathbf{b}) \cdot x_i^{(\ell)} + \mathcal{C}(\mathbf{w}_{m+\ell}) \odot \mathbf{b} \\
=& \Big(\sum_{i\in[m]} \mathcal{C}(\mathbf{w}_i) \cdot x_i^{(\ell)} + \mathcal{C}(\mathbf{w}_{m+\ell})\Big) \odot \mathbf{b} = \sum_{i\in[m]} \Big(\mathbf{w}_i \cdot x_i^{(\ell)} \cdot G + \mathbf{w}_{m+\ell} \cdot G\Big) \odot \mathbf{b} \\
=& \sum_{i\in[m]} \Big((\mathbf{w}_i \cdot x_i^{(\ell)} + \mathbf{w}_{m+\ell}) \cdot G\Big) \odot \mathbf{b} = \mathcal{C}(\mathbf{w}^{(\ell)}) \odot \mathbf{b},
\end{aligned}$$

where G is the generator matrix of the code $\mathcal{C}$ in standard form.

Now we prove the security starting with the case of a corrupt sender $\mathrm{P_S^*}$. We describe a simulator, $\mathcal{S}$, that interacts with the functionality $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$, such that no environment $\mathcal{Z}$ can distinguish between an interaction with $\mathcal{S}$ and $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ and an interaction with $\mathrm{P_S^*}$ and the protocol N-ROT. After describing the simulator we then argue indistinguishability of the real and ideal worlds.

Simulation for corrupt sender $\mathrm{P_S^*}$:

- $\mathcal{S}$ runs **Init** honestly, receiving $b_j \in \mathbb{F}_2$ for $j \in [n_C]$ from $\mathrm{P_S^*}$. It runs an internal copy of $\mathcal{F}_{\mathsf{ROT}}^{\kappa,n_C}$, sending random values $\mathbf{r}_{b_j}^j$, $j \in [n_C]$, to $\mathrm{P_S^*}$.
- In the **Extend** phase, $\mathcal{S}$ computes $\mathbf{t}_{b_j}^j$ using the internal values $(\mathbf{r}_{b_j}^j)$, $j \in [n_C]$. Then it samples uniformly random $\mathbf{u}^j, j \in [n_C]$, sends these values to $\mathrm{P_S^*}$, and also compute $\mathbf{q}^j$ as $\mathrm{P_S^*}$ would.
- In the consistency check step, $\mathcal{S}$ receives $x^{(\ell)} \in {\mathbb{F}_2}^m$, $\ell \in [s]$, from $\mathrm{P_S^*}$, then computes $\mathbf{q}^{(\ell)}$ as $\mathrm{P_S^*}$ would, and samples random $\mathbf{w}^{(\ell)} \in \mathbb{F}_2^{k_C}$. It computes $\mathbf{t}^{(\ell)} = \mathcal{C}(\mathbf{w}^{(\ell)}) \odot \mathbf{b} + \mathbf{q}^{(\ell)}$ and sends $(\mathbf{t}^{(\ell)}, \mathbf{w}^{(\ell)})$ for $\ell \in [s]$.
- In the **Output** step, output whatever $\mathrm{P_S^*}$ outputs and halt.

Now we argue indistinguishability. The values $\mathbf{r}_{b_j}^j$ obtained by $\mathrm{P_S^*}$ are identically distributed in the ideal and real processes because they are uniformly random in both cases. The real world $\mathbf{u}^j$ values are computationally indistinguishable from the simulated random values, since each $\mathbf{t}_{1-b_j}^j$ is a fresh output of PRG that is independent of the view of $\mathcal{Z}$, so hides the receiver's input $\mathbf{c}^j$. During the check $\mathrm{P_S^*}$ receives $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(s)}$ and $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(s)}$. The real world random values $\mathbf{w}_{m+\ell}$ act as a one-time pad to perfectly hide $\sum_{i\in[m]} \mathbf{w}_i \cdot x_i^{(\ell)}$, $\forall \ell \in [s]$, and the simulated $\mathbf{t}^{(\ell)}$ values are constructed so that the check always passes, so these values are identically distributed in both worlds.

Now we describe a simulator $\mathcal{S}$ in the case of a corrupt receiver $\mathrm{P_R^*}$.

Simulation for corrupt receiver $\mathrm{P_R^*}$:

- $\mathcal{S}$ runs **Init** honestly, it emulates $\mathcal{F}_{\mathsf{ROT}}^{\kappa,n_C}$ and forwards $(\mathbf{r}_0^j, \mathbf{r}_1^j)$, $j \in [n_C]$ to $\mathrm{P_R^*}$.

- During the **Extend** phase $\mathcal{S}$ receives the values $\mathbf{u}^j$ for $j \in [n_C]$ from $\mathrm{P}_{\mathrm{R}}^*$. Using the internal values $\mathbf{t}_0^j, \mathbf{t}_1^j$, the simulator reconstructs the matrices C and Q.
  In the check step, $\mathcal{S}$ samples random $x^{(1)}, \ldots, x^{(\ell)}$ and sends these values to $\mathrm{P}_{\mathrm{R}}^*$, then receives back $\mathbf{t}_*^{(\ell)}$ and $\mathbf{w}_*^{(\ell)}$ and runs the consistency check. If the check fails then send Abort to the functionality and halt.
- If the check passes, then for each row $\mathbf{c}_i$ of C, $\mathcal{S}$ decodes a message $\mathbf{w}_i \in \mathbb{F}_2^{k_C}$ (as described in Proposition 2).
  Then the simulator answers the random oracle queries:
  - If the query is $\mathsf{H}(i, \mathbf{q}_i + \mathcal{C}(\mathbf{w}_i) \odot \mathbf{b})$, send $(\mathsf{SenderOutput}, \mathbf{w}_i, i)$ to $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ and receive back $\mathbf{v}_{w_i,i}$. Return $\mathbf{v}_{w_i,i}$.
  - Otherwise, return a random value from $\mathbb{F}_2^\kappa$, consistent with previous queries.

Note that the simulation of the random oracle queries is *optimistic*, as it does not abort when queried on two conflicting inputs (e.g. corresponding to two different choices $\mathbf{w}_i$ and $\mathbf{w}_i'$). This is needed to ensure the security bound does not depend on N, and we shall return to this later.

Now we describe how $\mathcal{S}$ defines the corrupt $\mathrm{P}_{\mathrm{R}}^*$'s inputs which are sent to $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$, and prove indistinguishability. The **Init** step is perfectly simulated.

Let $\mathsf{PassCheck}$ denote the event that the consistency check passes for a cheating $\mathrm{P}_{\mathrm{R}}^*$. Since the simulator performs the consistency check in exactly the same way as an honest P_{S}, clearly $\Pr[\mathsf{PassCheck}]$ is the same in both worlds. If $\mathrm{P}_{\mathrm{R}}^*$ behaved honestly during the check then they should send the values

$$\begin{aligned}\mathbf{t}^{(\ell)} &= \sum_{i\in[m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell}\\ \mathbf{w}^{(\ell)} &= \sum_{i\in[m]} \mathbf{w}_i \cdot x_i^{(\ell)} + \mathbf{w}_{m+\ell},\end{aligned}$$

for some $\mathbf{w}_i$ values defining their inputs. Denote the values *actually* sent by $\mathrm{P}_{\mathrm{R}}^*$ by $\mathbf{t}_*^{(\ell)}$ and $\mathbf{w}_*^{(\ell)}$. Now, define $\mathbf{c}_*^{(\ell)} = \sum_{i\in[m]} \mathbf{c}_i \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell}^*$, and the differences

$$\begin{aligned}\overline{\mathbf{t}}^{(\ell)} &= \mathbf{t}^{(\ell)} + \mathbf{t}_*^{(\ell)}\\ \overline{\mathbf{c}}^{(\ell)} &= \mathbf{c}_*^{(\ell)} + \mathcal{C}(\mathbf{w}_*^{(\ell)}).\end{aligned} \tag{4}$$

The first value corresponds to the deviation in the value $\mathbf{t}_*^{(\ell)}$ that $\mathrm{P}_{\mathrm{R}}^*$ sent in the check, compared with what they should have sent, whilst the second corresponds to the deviation in the *codeword* defined by the $\mathbf{w}_*^{(\ell)}$ value that was sent, compared with the $\mathbf{c}_i$ values defined by the previous part of the protocol.

Now for the consistency check, $\mathcal{S}$ computes $\mathcal{C}(\mathbf{w}_*^{(\ell)})$ and $\mathbf{q}^{(\ell)}$, and checks that

$$\mathbf{q}^{(\ell)} + \mathbf{t}_*^{(\ell)} = \mathcal{C}(\mathbf{w}_*^{(\ell)}) \odot \mathbf{b}, \quad \forall \ell \in [s]. \tag{5}$$

Recall that $\mathbf{c}_i$ are the actual rows of the matrix C used by $\mathrm{P}_{\mathrm{R}}^*$ and that

$$\mathbf{q}^{(\ell)} = \sum_{i\in[m]} \mathbf{q}_i \cdot x_i^{(\ell)} + \mathbf{q}_{m+\ell} = \Big(\sum_{i\in[m]} \mathbf{c}_i \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell}^*\Big) \odot \mathbf{b} + \Big(\sum_{i\in[m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell}\Big).$$

Using this and the errors defined in (4), we can rewrite (5) as follows:

$$\begin{aligned}\mathbf{q}^{(\ell)} + \mathbf{t}_*^{(\ell)} &= \Big(\sum_{i\in[m]} \mathbf{c}_i \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell}^*\Big) \odot \mathbf{b} + \Big(\sum_{i\in[m]} \mathbf{t}_i \cdot x_i^{(\ell)} + \mathbf{t}_{m+\ell}\Big) + \mathbf{t}_*^{(\ell)} \\ &= \mathbf{c}_*^{(\ell)} \odot \mathbf{b} + \bar{\mathbf{t}}^{(\ell)},\end{aligned}$$

and combining this with (5) shows that passing the check is equivalent to $\mathrm{P}_{\mathrm{R}}^*$ choosing deviations $\bar{\mathbf{t}}^{(\ell)}, \bar{\mathbf{c}}^{(\ell)}$ such that:

$$\bar{\mathbf{t}}^{(\ell)} = \bar{\mathbf{c}}^{(\ell)} \odot \mathbf{b}. \tag{6}$$

Now, note that if $\bar{\mathbf{c}}^{(\ell)}[j] = 0$ for some $j \in [n_{\mathcal{C}}]$ then $\mathsf{PassCheck}$ implies that $\bar{\mathbf{t}}^{(\ell)}[j] = 0$ also. On the other hand, if $\bar{\mathbf{c}}^{(\ell)}[j] = 1$ then the check passes if and only if $\bar{\mathbf{t}}^{(\ell)}[j] = \mathbf{b}[j]$. This means that to pass the check, $\mathrm{P}_{\mathrm{R}}^*$ must guess one bit of $\mathbf{b}$ for each deviation in $\bar{\mathbf{c}}^{(\ell)}$.

Define the set of these deviation indices by $E^{(\ell)} = \{j \in [n_{\mathcal{C}}] : \bar{\mathbf{c}}^{(\ell)}[j] = 1\}$, and let $E = \bigcup_\ell E^{(\ell)}$. Also, let $S_{\mathbf{b}}$ be the set of all possible values of $\mathbf{b} \in \mathbb{F}_2^{n_{\mathcal{C}}}$ for which the messages sent by $\mathrm{P}_{\mathrm{R}}^*$ during the protocol will cause the check to pass. From the above, we have that for each $j \in E$, there is only one possibility for $\mathbf{b}[j]$, if $\mathbf{b} \in S_{\mathbf{b}}$. For $j \notin E$, passing the check is independent of $\mathbf{b}[j]$ so the check will pass for all $\mathbf{b}[j] \in \{0, 1\}$. This is summarized in the following proposition.

Proposition 1. *Let E and $S_{\mathbf{b}}$ be as above. Then if the check passes, it holds that*

- $|S_{\mathbf{b}}| = 2^{n_{\mathcal{C}} - |E|}$
- *For all $j \in E$ and every pair $\mathbf{b}_1, \mathbf{b}_2 \in S_{\mathbf{b}}$, $\mathbf{b}_1[j] = \mathbf{b}_2[j]$.*

In particular this means that, taking the probability over the sender's random choice of $\mathbf{b} \in \mathbb{F}_2^{n_{\mathcal{C}}}$,

$$\Pr[\mathsf{PassCheck}] = |S_{\mathbf{b}}|/2^{n_{\mathcal{C}}} = 2^{-|E|}. \tag{7}$$

The following proposition shows how to define the inputs of $\mathrm{P}_{\mathrm{R}}^*$, which are sent to $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$.

Proposition 2. *If the check passes then with probability at least $1 - 2^{-s} - 2^{-d_{\mathcal{C}}}$, $\mathcal{S}$ can extract values $\mathbf{w}_i \in \mathbb{F}_2^{k_{\mathcal{C}}}, \mathbf{e}_i \in \mathbb{F}_2^{n_{\mathcal{C}}}$, for $i \in [m]$, such that*

1. $\mathbf{c}_i = \mathcal{C}(\mathbf{w}_i) + \mathbf{e}_i$
2. $\mathbf{e}_i[j] = 0$ *for all* $j \notin E$.

Proof. We will use the following result on random linear combinations of codewords.

Lemma 1. *Let $\mathcal{C}$ be a linear code of length $n_{\mathcal{C}}$, $m' = m + s$ be an integer and $\mathbf{c}_i \in \mathbb{F}_2^{n_{\mathcal{C}}}$, for $i \in [m']$, such that there exists at least one $j \in [m]$ with $\mathbf{c}_j \notin \mathcal{C}$. Then, if $x_i^{(\ell)} \xleftarrow{\$} \mathbb{F}_2$, we have that:*

$$\Pr\Big(\forall \ell \in [s], \sum_{i\in[m]} \mathbf{c}_i \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell} \in \mathcal{C}\Big) \leq 2^{-s}.$$

Proof. Fix $\ell \in [s]$. For $t \leq m$, let E_t be the event that $\sum_{i\in[t]} \mathbf{c}_i \cdot x_i^{(\ell)} + \mathbf{c}_{m+\ell} \in \mathcal{C}$. Now if $\mathbf{c}_{t+1} \in \mathcal{C}$ then E_{t+1} occurs if and only if E_t does, since $\mathcal{C}$ is linear. In this case, we have $\Pr[E_{t+1}] = \Pr[E_t]$. On the other hand, if $\mathbf{c}_{t+1} \notin \mathcal{C}$ then there are two possible ways that E_{t+1} may happen: firstly, if E_t occurs and $x_{t+1} = 0$; secondly, if E_t does not occur and $x_{t+1} = 1$ then E_{t+1} may occur (depending on the $\mathbf{c}_i$). So in this case, we have $\Pr[E_{t+1}] \leq \Pr[E_t] \cdot \frac{1}{2} + (1 - \Pr[E_t]) \cdot \frac{1}{2} = \frac{1}{2}$. By the assumption that $\mathbf{c}_j \notin \mathcal{C}$ for some j, the second case must occur at least once, giving $\Pr[E_m] \leq \frac{1}{2}$. Since each of the s tests are independent, the overall probability is $\leq 2^{-s}$. □

For extraction of $\mathrm{P}_{\mathrm{R}}^*$'s inputs, note that it follows from (7) that if the check passes then $|E| < d_{\mathcal{C}}$, except with probability $2^{-d_{\mathcal{C}}}$. Assume that this holds. For a vector $\mathbf{c} \in \mathbb{F}_2^{n_{\mathcal{C}}}$, define $\mathbf{c}_{-E}$ to be the vector obtained by removing the positions $j \in E$. Let $\mathcal{C}_{-E}$ be the *punctured code* consisting of the codewords $\{\mathbf{c}_{-E} : \mathbf{c} \in \mathcal{C}\}$.

By definition of E, we must have $\overline{\mathbf{c}}_{-E}^{(\ell)} = \mathbf{0}$, and because $\mathbf{c}_*^{(\ell)} = \overline{\mathbf{c}}^{(\ell)} + \mathcal{C}(w_*^{(\ell)})$, we also have $(\mathbf{c}_*^{(\ell)})_{-E} \in \mathcal{C}_{-E}$, for every $\ell \in [s]$. Therefore, by applying Lemma 1 with the code $\mathcal{C}_{-E}$ and vectors $(\mathbf{c}_1)_{-E}, \ldots, (\mathbf{c}_{m'})_{-E}$, it holds that for every $i \in [m]$, $(\mathbf{c}_i)_{-E} \in \mathcal{C}_{-E}$, except with probability $\leq 2^{-s}$. Since $|E| < d_{\mathcal{C}}$, where $d_{\mathcal{C}}$ is the minimum distance of $\mathcal{C}$, $\mathcal{C}_{-E}$ has minimum distance at least 1, so for each $i \in [m]$ we can decode a value $\mathbf{w}_i \in \mathbb{F}_2^{k_{\mathcal{C}}}$ such that $(\mathbf{c}_i)_{-E} = \mathcal{C}_{-E}(\mathbf{w}_i)$. This implicitly defines errors $\mathbf{e}_i \in \mathbb{F}_2^{n_{\mathcal{C}}}$, which satisfy both parts of the proposition.

□

Next, we consider the random oracle queries made by the environment, $\mathcal{Z}$. Whenever $\mathcal{Z}$ sends a query of the form $(i, \mathbf{q}_i + \mathcal{C}(\mathbf{w}_i) \odot \mathbf{b})$, $\mathcal{S}$ responds with the correct output from $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$. Note that since $\mathbf{q}_i = \mathbf{t}_i + \mathbf{c}_i \odot \mathbf{b}$, we have

$$\begin{aligned}\mathbf{q}_i + \mathcal{C}(\mathbf{w}_i) \odot \mathbf{b} &= \mathbf{t}_i + (\mathbf{c}_i + \mathcal{C}(\mathbf{w}_i)) \odot \mathbf{b} \\ &= \mathbf{t}_i + \mathbf{e}_i \odot \mathbf{b}.\end{aligned}$$

Since we have already shown that the value of $\mathbf{e}_i \odot \mathbf{b}$ must be known to $\mathrm{P}_{\mathrm{R}}^*$ for the check to have passed, $\mathcal{Z}$ cannot learn any additional information from making such a query.

On the other hand, if $\mathcal{Z}$ sends a query of the form

$$(i, \mathbf{q}_i + \mathcal{C}(\mathbf{w}') \odot \mathbf{b},) \quad \text{for some } i \in [m], \mathbf{w}' \neq \mathbf{w}_i$$

then $\mathcal{Z}$ *may* be able to distinguish by comparing the response with the honest sender's output; but this only works if the corresponding choice $\mathbf{w}'$ was requested by P_{S}. Let $\mathsf{BadQuery}$ be the event that this happens. Note that if $\mathsf{BadQuery}$ does not occur, then Proposition 2 implies that the real and simulated executions are identically distributed in the view of $\mathcal{Z}$, except with probability $2^{-s} + 2^{-d_{\mathcal{C}}}$ (if $\mathcal{S}$ fails to decode the inputs). We now show that there is only a negligible probability of $\mathsf{BadQuery}$ happening.

Again using Proposition 2, we can rewrite such a critical query as

$$(i, \mathbf{t}_i + (\mathcal{C}(\mathbf{w}') + \mathcal{C}(\mathbf{w}_i) + \mathbf{e}_i) \odot \mathbf{b}),$$

where $\mathbf{w}' \neq \mathbf{w}_i$ and $\mathbf{e}_i[j] = 0$ for all $j \notin E$. Since $\mathbf{t}_i$ and $(\mathbf{b}[j])_{j \in E}$ are already known to $\mathrm{P}_{\mathrm{R}}^*$ (by Proposition 1), coming up with such a query is at least as hard as computing

$$\big((\mathcal{C}(\mathbf{w}') + \mathcal{C}(\mathbf{w}_i)) \odot \mathbf{b}\big)_{-E}, \tag{8}$$

for some $\mathbf{w}' \neq \mathbf{w}_i$. For each such $\mathbf{w}'$, we have $\mathsf{wt}_{\mathsf{H}}((\mathcal{C}(\mathbf{w}') + \mathcal{C}(\mathbf{w}_i))_{-E}) \geq d_{\mathcal{C}} - |E|$, so for fixed values of $\mathbf{w}', \mathbf{w}_i$, the probability of guessing (8) is $\leq 2^{|E|-d_{\mathcal{C}}}$. Taking a union bound over all $|S_i|$ possible values of $\mathbf{w}'$ corresponding to an output of P_{S} gives probability at most $(|S_i| - 1) \cdot 2^{|E|-d_{\mathcal{C}}}$ per query. [3] If $\mathrm{P}_{\mathrm{R}}^*$ makes q queries to the random oracle and we define $N' = \max_i |S_i|$ to be the maximum number of outputs the sender learns in any OT, then in total we get

[3] We do not need to also take the union bound over $\mathbf{w}_i$, because i is contained in each query and therefore fixed.

$$\Pr[\mathsf{BadQuery}] \leq q \cdot (N' - 1) \cdot 2^{|E|-d_C}.$$

Combining this with (7) and the failure event (which we denote $\mathsf{DecodeFail}$) from Proposition 2, we see that the overall distinguishing advantage of the environment is no more than

$$\begin{aligned}\Pr[\mathsf{PassCheck} \wedge (\mathsf{DecodeFail} \vee \mathsf{BadQuery})] &\leq 2^{-|E|} \cdot (2^{-s} + 2^{-d_C} + q \cdot (N' - 1) \cdot 2^{|E|-d_C}) \\ &= q \cdot (N' - 1) \cdot 2^{-d_C} + 2^{-s-|E|} + 2^{-d_C-|E|}\,,\end{aligned}$$

which is negligible in s and κ if $d_C \geq \kappa$.

□

Instantiating the Code. It remains to instantiate the binary linear code, $\mathcal{C}$, to obtain a 1-out-of-N random OT protocol for a desired power of two choice of N. As well as the repetition code (for 1-out-of-2 OT), we suggest a more efficient form of the Walsh-Hadamard code for $N \leq 512$; a binary Golay code for $N = 2048$; and BCH codes for values of N that are exponential in the security parameter. The parameters of these codes are presented in Table 1; note that the code length determines exactly the amount of communication required per extended OT. We obtained the generator matrices for all of these codes using Sage [4].

Recall that $\mathcal{C}$ is an $[n_C, k_C, d_C]$ binary linear code, and the protocol requires $k_C = \log_2 N$ and $d_C \geq \kappa$ to achieve computational security parameter κ.

Repetition Code. If $N = 2$ then the repetition code $\mathcal{C}_{\mathsf{rep}} = \{0^\kappa, 1^\kappa\}$ gives a 1-out-of-2 OT extension protocol that is very similar to the previous, most efficient protocol [KOS15] (our consistency check is slightly more expensive, but still has negligible cost for many OTs).

Walsh-Hadamard Codes. The Walsh-Hadamard code of dimension $k_C = \log_2 N$ is defined by taking the inner product (over $\mathbb{F}_2$) of the message with all N length-k_C vectors:

$$\mathcal{C}_{\mathsf{WH}}(x) = (\langle x, y \rangle)_{y \in \{0,1\}^{k_C}}$$

This has length N and minimum distance $d_C = N/2$, so is an $[N, \log_2 N, N/2]$ code.

A more efficient choice is the *punctured Walsh-Hadamard code*, which restricts y to be one in the first position:

$$\mathcal{C}^*_{\mathsf{WH}}(x) = (\langle x, y \rangle)_{y \in \{1\} \times \{0,1\}^{k_C - 1}}$$

This has length $N/2$ and can be shown to have minimum distance $N/4$, so is an $[N/2, \log_2 N, N/4]$ code. These are slightly better parameters than the standard W-H code.

To use $\mathcal{C}^*_{\mathsf{WH}}$ for 1-out-of-N OT we simply need to ensure that $d_C = N/4 \geq \kappa$. So, fixing $N = 512$ gives us a $[256, 9, 128]$ code, allowing 1-out-of-512 OTs at a cost of sending just 256 bits, with 128-bit security. [5]

[4] `http://www.sagemath.org`

[5] The punctured W-H code provides no benefit for $N > 512$, since this would cost at least 512 bits of communication, and building 1-out-of-N OT from smaller 1-out-of-N' OTs (similar to [NP99]) would be more efficient.

Code	N	Length	Distance/Security
Repetition [IKNP03]	2	128	128
Walsh-Hadamard [KK13]	≤ 256	256	128
Punctured Walsh-Hadamard	≤ 512	256	128
Binary Golay	2048	384	128
BCH-511	2^{76}	511	171
BCH-1023	2^{443}	1023	128

Table 1: Parameters for various choices of code

Smaller values of N. If $N < 4\kappa$ then we can repeat the $[N/2, \log_2 N, N/4]$ punctured W-H code $4\kappa/N$ times to obtain a code of length 2κ and distance $N/4 \cdot 4\kappa/N = \kappa$, as required for security. So, with $\kappa = 128$ this has the same cost as 1-out-of-512 OT, for any power of two $N \leq 512$.

BCH codes. For larger values of N, BCH codes can be used to give much better parameters than Walsh-Hadamard codes.

For example, the $[511, 76, 171]$ BCH code would allow $N = 2^{76}$, with 171-bit security. Since in practice n will be much smaller than this, the code can be shortened to get parameters $[511 - c, 76 - c, \geq 171]$, and hence $N = 2^{76-c}$, for any desired choice of c.

Binary Golay code. The binary Golay code is a $[24, 12, 8]$ binary linear code, often used in communications. Extending this code (with repetition) 16 times gives a $[384, 12, 128]$ code, which can be used for 1-out-of-2048 OT, at the cost of sending 384 bits, with 128 bits of security. This improves upon both the Walsh-Hadamard and BCH instantiations for the same size of N.

Non Two-Power Values of n. If n is not a power of two, then we run into the problem that no binary code $\mathcal{C}$ can contain exactly n codewords. If $\mathcal{C}$ is chosen to have more than N, then the receiver may choose a value $w_i \notin \{0, \ldots, n-1\}$, which still defines a valid codeword. The consistency check will still go through, but at the end of the protocol the receiver will not obtain a valid output. However, in many applications this may not pose a problem, and the OT functionality can easily be modified to model this possibility.

4 Security in the Standard Model

In this section we consider the case of non-random 1-out-of-N OT. In this protocol (Fig. 3), we remove the random oracle assumption and prove security in the standard model. Similarly to [ALSZ15], we need a stronger version of correlation robustness than that given in [IKNP03], and require that the secret correlation $\mathbf{b}$ comes from a distribution of min-entropy k and in addition is multiplied by a codeword in the binary linear code $\mathcal{C}$.

For self-containment we recall the definitions of correlation robustness and strong correlation robustness in Appendix A.

Definition 1 (k-min-entropy code correlation robustness). *Let χ be a distribution on $\mathbb{F}_2^{n_\mathcal{C}}$ with min-entropy k and $\mathcal{C}$ be an $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]$ binary linear code. An efficiently computable function $\mathsf{H} : \mathbb{F}_2^{n_\mathcal{C}} \to \mathbb{F}_2^\kappa$ is said to be k-min-entropy $\mathcal{C}$-*correlation robust *if it holds that:*

$$\{\mathbf{t}_i, \mathsf{H}(\mathbf{t}_i + \mathbf{c} \odot \mathbf{b})\}_{i \in [m], \mathbf{c} \in \mathcal{C}} \stackrel{c}{\equiv} \mathcal{U}^{m \cdot n_\mathcal{C} + (m + |\mathcal{C}|) \cdot \kappa},$$

Protocol N-$\mathsf{OT}^{\kappa,m}$ (Standard Model)

INPUT OF $\mathrm{P_S}$: m tuples $(\mathbf{v}_{0,i}, \ldots, \mathbf{v}_{N-1,i})$ of κ-bit strings, $1 \leq i \leq m$.
INPUT OF $\mathrm{P_R}$: m selection integers $(w_1, \ldots, w_m)$, such that $0 \leq w_i < N-1 \ \ \forall i \in [m]$, and encoded as bit strings $\mathbf{w}_1, \ldots, \mathbf{w}_m \in \mathbb{F}_2^{\kappa}$.
COMMON INPUT: κ and s are the computational and the statistical security parameter; an $[n_{\mathcal{C}}, k_{\mathcal{C}}, d_{\mathcal{C}}]$ binary code $\mathcal{C}$ such that $n_{\mathcal{C}} \geq \kappa, \ \ k_{\mathcal{C}} \geq \log N, \ \ d_{\mathcal{C}} = \kappa$.
REQUIRE: $\mathsf{H} : \mathbb{F}_2^{n_{\mathcal{C}}} \to \mathbb{F}_2^{\kappa}$ a k-min-entropy strongly $\mathcal{C}$-correlation-robust and $\mathsf{PRG} : \{0,1\}^{\kappa} \to \{0,1\}^{m'}$.

Init: As in Protocol N-$\mathsf{ROT}^{\kappa,m}$.
Extend: As in Protocol N-$\mathsf{ROT}^{\kappa,m}$.
Output: $\mathrm{P_S}$ sends, $\forall i \in [m]$ and $0 \leq w < N-1$: $\mathbf{y}_{w,i} = \mathbf{v}_{w,i} + \mathsf{H}(\mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})$. $\mathrm{P_R}$ recovers, $\forall i \in [m]$: $\mathbf{v}_{w_i,i} = \mathbf{y}_{w_i,i} + \mathsf{H}(\mathbf{t}_i)$.

Fig. 3: An implementation of $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$ extending $\mathcal{F}_{2\text{-}\mathsf{OT}}^{\kappa,n_{\mathcal{C}}}$ in the Standard Model.

where $\mathbf{b} \xleftarrow{\$} \chi$ *and* $\mathbf{t}_1, \ldots, \mathbf{t}_m \in \mathbb{F}_2^{n_{\mathcal{C}}}$ *are independent and uniformly distributed.*

Similarly, $\mathsf{H}(\cdot)$ *is said to be* k*-min-entropy* strongly $\mathcal{C}$-correlation robust *if it holds that:*

$$\{\mathsf{H}(\mathbf{t}_i + \mathbf{c} \odot \mathbf{b})\}_{i \in [m], \mathbf{c} \in \mathcal{C}} \stackrel{c}{\equiv} \mathcal{U}^{(m+|\mathcal{C}|)\cdot\kappa},$$

where $\mathbf{b} \xleftarrow{\$} \chi$*, for any distribution of the* $\{\mathbf{t}_i\}_{i \in [m]}$*.*

Notice that in the values used to mask $\mathrm{P_S}$'s inputs, it is the receiver that effectively chooses the $\mathbf{t}_j$'s, and they can not only choose these values non-uniformly, but even maliciously. This is the reason why we need a *strong* code-correlation robust hash function.

We claim that if H is a k-min-entropy strongly correlation robust function for all $n_{\mathcal{C}} - s \leq k \leq n_{\mathcal{C}}$, then the protocol is secure in the standard model. For further discussion on parameter choices regarding this assumption, see Section 4.1.

Theorem 2. *Assuming that* H *is* k*-min-entropy strongly code-correlation robust for all* $k \in \{n_{\mathcal{C}} - s, \ldots, n_{\mathcal{C}}\}$*, and* PRG *is a pseudo-random generator, the protocol* N-$\mathsf{OT}^{\kappa,m}$ *in Fig. 3 securely implements* $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$ *in the* $\mathcal{F}_{2\text{-}\mathsf{OT}}$*-hybrid model against a static malicious adversary.*

The high level idea of the proof is the following. Assuming that there exists $\mathcal{Z}$ that can distinguish between the ideal and real execution of the protocol with significant advantage δ, that such $\mathcal{Z}$ can be used to distinguish between the two distributions in the strong code-correlation robustness definition. More precisely, we prove the following porposition.

Proposition 3. *Suppose there exists an adversary* $\mathcal{A}$ *and distinguisher* $\mathcal{Z}$ *such that* $\mathcal{Z}$ *breaks UC security of the protocol with advantage* $\mathrm{Adv}(\mathcal{Z}) = \delta$*, running in time* t*. Then there exists* $k \geq n_{\mathcal{C}} - s$ *and a distinguisher* $\mathcal{D}$ *that runs in time* t *and breaks the* k*-min-entropy correlation robustness of* H *with advantage* $2^{n_{\mathcal{C}}-k} \cdot \delta$*.*

In an asymptotic sense, this means that if we let $s = c \cdot \kappa$ for some constant $c \in (0,1)$, then any adversary breaking security of the protocol with advantage δ in time $t = \mathsf{poly}(\kappa)$ can be transformed into a distinguisher that breaks H for some $k \geq (1-c) \cdot \kappa$ with advantage $2^{n_{\mathcal{C}}-k} \cdot \delta$, running in time $O(t) = \mathsf{poly}(k)$.

In concrete terms, if we fix $\kappa = d_{\mathcal{C}} = 128$ and $s = 40$, then the protocol is secure if for all $88 \leq k \leq 128$, there are no attacks on the min-entropy code-correlation robustness of H that succeed with probability significantly larger than 2^{-k}.

Proof. The case of a corrupt sender is the same as in the proof of Theorem 1. To prove Theorem 2 for the case of a corrupt receiver, $\mathrm{P}_{\mathrm{R}}^{*}$, we will describe a simulator, $\mathcal{S}$, and show that any environment $\mathcal{Z}$ who can distinguish between the real and ideal executions of the protocol can be used to construct a distinguisher which breaks the correlation robustness assumption on H.

$\mathcal{S}$ begins by simulating the protocol as before in the proof of Theorem 1. If the check passes, $\mathcal{S}$ decodes the receiver's inputs $\mathbf{w}_i \in \mathbb{F}_2^{\kappa}$ and the errors $\mathbf{e}_i$, $\forall i \in [m]$, as in the proof of Theorem 1, and computes the messages:

$$\begin{aligned}\mathbf{y}_{\mathbf{w}_i,i} &= \mathbf{v}_{\mathbf{w}_i,i} + \mathsf{H}(\mathbf{t}_i + \mathbf{e}_i \odot \mathbf{b})\\ \mathbf{y}_{\mathbf{w},i} &\xleftarrow{\$} \mathbb{F}_2^{n_C}, \quad \forall \mathbf{w} \neq \mathbf{w}_i,\end{aligned}$$

where $\mathbf{v}_{\mathbf{w}_i,i}$ is the i-th receiver's output obtained from $\mathcal{F}_{N\text{-}\mathsf{OT}}^{\kappa,m}$.

Note that the simulated $\mathbf{y}_{\mathbf{w}_i,i}$ values are computed exactly as in the real world. Conversely, all other $\mathbf{y}_{\mathbf{w},i}$ values (for $\mathbf{w} \neq \mathbf{w}_i$) in the ideal execution are uniformly random, whilst in the real world the sender computes these values as:

$$\begin{aligned}\mathbf{y}_{\mathbf{w},i} &= \mathbf{v}_{\mathbf{w},i} + \mathsf{H}(\mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})\\ &= \mathbf{v}_{\mathbf{w},i} + \mathsf{H}(\mathbf{t}_i + (\mathcal{C}(\mathbf{w}_i) + \mathcal{C}(\mathbf{w}) + \mathbf{e}_i) \odot \mathbf{b})\end{aligned} \tag{9}$$

for every $\mathbf{w} \neq \mathbf{w}_i$ and $i \in [m]$.

We now show that no environment $\mathcal{Z}$ can distinguish between the real and ideal process. Using the standard UC notation, for an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$, we let

$$\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(1^{\kappa}, 1^{s}, (\mathbf{v}_{0,i}, \ldots, \mathbf{v}_{N-1,i})_{i\in[m]}, (w_i)_{i\in[m]})$$

denote the random variable describing the output of $\mathcal{Z}$ in a real execution of Π with adversary $\mathcal{A}$ and input $((\mathbf{v}_{0,i}, \ldots, \mathbf{v}_{N-1,i})_{i\in[m]}, (w_i)_{i\in[m]})$. Similarly, let

$$\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(1^{\kappa}, 1^{s}, (\mathbf{v}_{0,i}, \ldots, \mathbf{v}_{N-1,i})_{i\in[m]}, (w_i)_{i\in[m]})$$

denote the random variable describing the output of $\mathcal{Z}$ after interacting with the ideal execution with adversary $\mathcal{S}$, the functionality $\mathcal{F}$ and input $((\mathbf{v}_{0,i}, \ldots, \mathbf{v}_{N-1,i})_{i\in[m]}, (w_i)_{i\in[m]})$. For sake of simplicity, we omit the inputs and refer to these as $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ and $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, respectively. We assume the output of $\mathcal{Z}$ to be a single bit, considered as a guess at one of the two executions REAL or IDEAL. The advantage of $\mathcal{Z}$ is then given by:

$$\mathrm{Adv}(\mathcal{Z}) := |\Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} = 1] - \Pr[\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = 1]|.$$

Similarly, for an event X that may occur in both executions, we define the conditional advantage as:

$$\mathrm{Adv}(\mathcal{Z}|X) := |\Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} = 1|X] - \Pr[\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = 1|X]|.$$

Now, suppose there exists a set of inputs for which $\mathcal{Z}$ distinguishes between the two worlds with noticeable advantage, δ. Note that the probability of the consistency check failing (i.e. $\mathbf{b} \notin S_{\mathbf{b}}$) is the same in both worlds, and if this happens then the simulation is perfect, so we have

$$\mathrm{Adv}(\mathcal{Z}|\mathbf{b} \notin S_{\mathbf{b}}) = 0. \tag{10}$$

On the other hand, if the consistency check passes then using (7) and the rules of conditional probability, we have:

$$\begin{aligned}\delta &= |\Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} = 1] - \Pr[\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = 1]| \\ &= |\Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} = 1 | \mathbf{b} \in S_{\mathbf{b}}] \cdot 2^{-|E|} - \Pr[\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = 1 | \mathbf{b} \in S_{\mathbf{b}}] \cdot 2^{-|E|}| \\ &= 2^{-|E|} \cdot |\Pr[\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} = 1 | \mathbf{b} \in S_{\mathbf{b}}] - \Pr[\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = 1 | \mathbf{b} \in S_{\mathbf{b}}]| \\ &= 2^{-|E|} \cdot \mathrm{Adv}(\mathcal{Z} | \mathbf{b} \in S_{\mathbf{b}}). \end{aligned} \tag{11}$$

Now we show that such a $\mathcal{Z}$ can be used to distinguish between the two distributions in the strong code-correlation robustness definition. More precisely:

Lemma 2. *Suppose there exists an adversary $\mathcal{A}$ and a distinguisher $\mathcal{Z}$ such that $\mathcal{Z}$ breaks UC security of the protocol with advantage $\mathrm{Adv}(\mathcal{Z}) = \delta$. Then there exists $k \geq n_{\mathcal{C}} - s$ and a distinguisher $\mathcal{D}$, which breaks the k-min-entropy correlation robustness of H with advantage $2^{n_{\mathcal{C}}-k} \cdot \delta$.*

Proof. In a nutshell, $\mathcal{D}$ sets up a copy of $\mathcal{Z}$, then goes through the protocol with $\mathcal{Z}$ and uses its outputs to guess the distribution of a strongly $\mathcal{C}$-correlation robust challenge. We show that if $\mathcal{Z}$ guesses correctly between the real and ideal world execution then $\mathcal{D}$ guesses whether the challenge is uniformly distributed, or distributed as $\mathsf{H}(\mathbf{t}'_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})$, for some values $\mathbf{t}'_i$ provided to the challenger.

$\mathcal{D}$ starts the execution of the protocol in the same way the simulator would do, with some exceptions, as explained in the following. Fix the random coins of the adversary $\mathcal{A}$, and the values $x_i^{(\ell)}$ (provided by $\mathcal{D}$), to maximize the success probability of $\mathcal{Z}$. This uniquely determines the adversary's messages $\{\mathbf{r}_0^j, \mathbf{r}_1^j, \mathbf{u}^j\}_{j\in[n_C]}$ and $\{\mathbf{w}_*^{(\ell)}, \mathbf{t}_*^{(\ell)}\}_{\ell\in[s]}$, such that $\mathrm{Adv}(\mathcal{Z}) \geq \delta$ (ensuring that $\mathbf{b} \in S_{\mathbf{b}}$). Recall that $\{\mathbf{w}_*^{(\ell)}, \mathbf{t}_*^{(\ell)}\}_{\ell\in[s]}$ denote the values actually sent by $\mathrm{P}_{\mathrm{R}}^*$. As in the proof of Theorem 1, this transcript also uniquely defines an error set E and vectors $\mathbf{c}_i, \mathbf{e}_i \in \mathbb{F}_2^{n_C}, \mathbf{w}_i \in \mathbb{F}_2^{k_C}$ such that $\mathbf{c}_i = \mathcal{C}(\mathbf{w}_i) + \mathbf{e}_i$ and $|E| < s$ (except with negligible probability).

Let $k = n_{\mathcal{C}} - |E|$. The distinguisher $\mathcal{D}$ defines χ to be the distribution on $\mathbb{F}_2^{n_C}$ where each position $j \in E$ is fixed to be consistent with the adversary's transcript, and all k positions $j \notin E$ are chosen uniformly at random. This is precisely the uniform distribution on the set, $S_{\mathbf{b}}$, of all secrets $\mathbf{b}$ for which the adversary could pass the consistency check, and we have $H_\infty(\chi) = n_{\mathcal{C}} - |E|$. Next, $\mathcal{D}$ defines $\mathbf{t}'_i = \mathbf{t}_i + \mathbf{e}_i \odot \mathbf{b}$ (recall that $\mathbf{e}_i \odot \mathbf{b}$ is fixed for all $\mathbf{b} \in S_{\mathbf{b}}$ at this point, so this is well-defined) and sends $\chi, \{\mathbf{t}'_i\}_{i\in[m]}$ to the correlation robustness challenger. $\mathcal{D}$ receives back a set of challenge samples $\{\mathbf{x}_{\mathbf{w},i}\}_{i\in[m],\mathbf{w}\in\mathbb{F}_2^{k_C}}$ and must guess whether these are uniform or distributed as

$$\mathbf{x}_{\mathbf{w},i} = \mathsf{H}(\mathbf{t}'_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b}),$$

for some $\mathbf{b} \xleftarrow{\$} \chi$. Note that if the latter is true then we have $\mathbf{x}_{\mathbf{w},i} = \mathsf{H}(\mathbf{q}_i + (\mathcal{C}(\mathbf{w}_i) + \mathcal{C}(\mathbf{w})) \odot \mathbf{b})$, where $\mathbf{q}_i$ is defined as in the protocol. To match this up with the masks used in the final stage of the protocol (9), $\mathcal{D}$ permutes the i-th set of samples with the mapping $\mathbf{w} \mapsto \mathbf{w} + \mathbf{w}_i$, defining the final set of protocol messages sent to $\mathcal{Z}$ as:

$$\begin{aligned}\mathbf{y}_{\mathbf{w}_i,i} &= \mathbf{v}_{\mathbf{w}_i,i} + \mathsf{H}(\mathbf{t}'_i) \\ \mathbf{y}_{\mathbf{w},i} &= \mathbf{v}_{\mathbf{w},i} + \mathbf{x}_{\mathbf{w}+\mathbf{w}_i,i}, \quad \text{for all } \mathbf{w} \neq \mathbf{w}_i.\end{aligned}$$

Both in the simulation and real world execution, the view $\mathcal{Z}$ sees consists of the view of the corrupt party $\mathrm{P}_{\mathrm{R}}^*$, and until the last step (when $\mathrm{P}_{\mathrm{R}}^*$ receives the values $\mathbf{y}_{\mathbf{w},i}$), it essentially consists of its random tape. The view generated by $\mathcal{D}$, before the last step, is statistically indistinguishable from the view generated by $\mathcal{S}$ and the one produced in the real protocol. We can then distinguish two cases:

1. If $\mathbf{x}_{\mathbf{w},i}$ are uniformly distributed then the view produced by $\mathcal{D}$ is statistically indistinguishable from the view generated by $\mathcal{S}$. We denote by $\mathcal{D}_{\mathsf{IDEAL}}$ the random variable describing the output of $\mathcal{D}$ in this case.
2. Otherwise, if the values $\mathbf{x}_{\mathbf{w},i}$ are distributed as $\mathsf{H}(\mathbf{t}'_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})$, then we have $\mathbf{y}_{\mathbf{w},i} = \mathbf{v}_{\mathbf{w},i} + \mathsf{H}(\mathbf{q}_i + \mathcal{C}(\mathbf{w}) \odot \mathbf{b})$, so the outputs seen by $\mathcal{Z}$ are identical to those in a real run of the protocol and we denote by $\mathcal{D}_{\mathsf{REAL}}$ the random variable describing the output of $\mathcal{D}$.

The distinguishing advantage of $\mathcal{D}$ is therefore identical to that of $\mathcal{Z}$, conditioned on $\mathbf{b} \in S_{\mathbf{b}}$ (and ignoring the failure events from Proposition 2, which occur with negligible probability). So using (11), with overwhelming probability we have:

$$\mathrm{Adv}(\mathcal{D}) = \mathrm{Adv}(\mathcal{Z} | \mathbf{b} \in S_{\mathbf{b}}) = 2^{|E|} \cdot \delta = 2^{n_{\mathcal{C}} - k} \cdot \delta,$$

□

This also concludes the proof of Theorem 2. □

4.1 Parameter choices for code-correlation robustness

To conclude this section, we remark that our requirement of assuming code-correlation robustness for a *range* of parameters $k \in \{n_{\mathcal{C}} - s, \ldots, n_{\mathcal{C}}\}$ is slightly unusual. We could instead have proven security by assuming code-correlation robustness for a single choice of k (similarly to [ALSZ15]), but this would require setting $n_{\mathcal{C}}$ to at least $\kappa + s$, potentially increasing the communication and number of base OTs in the protocol. We believe that our assumption provides optimal efficiency in terms of the number of base OTs while still providing meaningful security, in both a theoretical sense and in giving concrete security guarantees for the protocol.

In an asymptotic sense, our theorem means that if we let the statistical security parameter $s = c \cdot \kappa$ for some constant $c \in (0, 1)$, then any adversary breaking security of the protocol with advantage δ can be transformed into a distinguisher that breaks H for some $k \geq (1 - c) \cdot \kappa$ with advantage $2^{n_{\mathcal{C}} - k} \cdot \delta$, running in roughly the same time.

In concrete terms, consider the 1-out-of-2 case with repetition coding, and parameters $\kappa = n_{\mathcal{C}} = d_{\mathcal{C}} = 128$ and $s = 40$. The protocol is secure if for all $88 \leq k \leq 128$, there are no attacks on the min-entropy code-correlation robustness of H that succeed with probability significantly larger than 2^{-k}. Importantly, this does not mean that the protocol offers only "88 bits of security", as our reduction actually *amplifies* (depending on k) the success probability of any adversary. For example, any attack on the protocol with $k = 88$ results in an attack against the correlation robust function that is 2^{40} *times more successful* than the attack on the protocol. Intuitively, therefore, these parameters are suitable for 128 bits of computational security.

5 Application to Private Set Intersection

We now show how to apply the 1-out-of-N random OT extension protocol to increase the efficiency and obtain stronger security guarantees in existing private set intersection (PSI) protocols. We describe a simpler and more efficient private set inclusion protocol with active security, which is used as a key component of the most efficient passively secure PSI protocols.

5.1 Private Set Inclusion

A core building block of OT-based PSI protocols is a *private set inclusion* protocol, where party P_A has input $a \in \{0,1\}^k$, party P_B has input a set $B \subset \{0,1\}^k$ and the parties wish to learn whether $a \in B$. Note that the special case of $|B| = 1$ is a private equality test.

The previous most efficient protocol [PSSZ15, Sec. 6.1] requires $t = k/8$ executions of 1-out-of-256 random OT, and uses the KK protocol with length 256 Walsh-Hadamard codes. However, with the observation that our random OT protocol can be used for exponentially large values of N, we can in fact choose $N = 2^k$ and perform a private set inclusion with just a single 1-out-of-N random OT. This is possible because the OT sender is only required to learn one of the random OT outputs in order to run the set inclusion protocol.

The protocol, shown in Fig. 5, is very simple: P_A inputs their value a as the receiver's choice in a 1-out-of-N random OT, and P_B inputs each of their values $b \in B$ to obtain $|B|$ of the sender's random outputs. Thus, P_A learns a random value r_a and P_B learns a set of random values $R = \{r_b\}_{b \in B}$. P_B randomly permutes R and sends this to P_A, who checks whether $r_a \in R$ to determine the result (and can send this to P_B if desired).

Since P_A only learns one of the random OT outputs initially, all other possible elements of the set R are uniformly random so do not leak any information on P_B's input. Note that because our 1-out-of-N OT protocol is actively secure, we actually obtain an actively secure private set inclusion protocol (although this does not seem to suffice to make the PSI protocol of [PSSZ15] actively secure).

Functionality $\mathcal{F}^k_{\mathsf{SetInc}}$

Inputs: P_A inputs $a \in \{0,1\}^k$ and P_B inputs $B \subset \{0,1\}^k$.
Output: If $a \in B$ then output $(1, |B|)$ to P_A. Otherwise, output $(0, |B|)$ to P_A. P_B receives no output.

Fig. 4: One-sided private set inclusion functionality

The ideal functionality we implement is given in Fig. 4; note that this also needs to output the size of P_B's set to P_A.

Theorem 3. *Protocol Π_{SetInc} securely realizes the functionality $\mathcal{F}^k_{\mathsf{SetInc}}$ with statistical security parameter s.*

Proof. Suppose P_A is corrupt. The simulator receives $a \in \{0,1\}^k$ as P_A's input to $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$, sends this to the $\mathcal{F}^k_{\mathsf{SetInc}}$ functionality and receives a bit y and the set size $|B|$. $\mathcal{S}$ samples a random $r_a \in \{0,1\}^k$ and sends this to P_A (acting as the $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ output). Next, if $y = 0$ then $\mathcal{S}$ sends a random subset $R \subset \{0,1\}^k$ of size $|B|$ to P_A. Otherwise, $\mathcal{S}$ constructs R to contain r_a in a random

Protocol Π_{SetInc}

Inputs: P_A has input $a \in \{0,1\}^k$ and P_B has input $B \subset \{0,1\}^k$.

1. The parties run $\mathcal{F}^{s,1}_{N\text{-}\mathsf{ROT}+}$ with $N = 2^k$, where P_A plays receiver with input a.
2. P_A receives $r_a \in \{0,1\}^s$
3. For each $b \in B$, P_B calls $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ with $(\mathsf{SenderOutput}, 0, b)$ to obtain r_b.
4. P_B randomly permutes and sends $R = \{r_b\}_{b \in B}$ to P_A.
5. P_A outputs 1 if $r_a \in R$ and 0 otherwise.

Fig. 5: Private set inclusion protocol

position and $|B| - 1$ random values elsewhere, and sends R. It is easy to see that the view of P_A is identically distributed to the view in a real execution. As for the output of P_A, notice that if $a \in B$ then both executions always output 1. If $a \notin B$ then the ideal execution always outputs $y = 0$, whilst the real execution outputs 1 with probability $\leq |B| \cdot 2^{-s}$, in case $r_a \in R$ by chance, so the two worlds are statistically indistinguishable.

Now consider a corrupted P_B. Whenever P_B queries $\mathcal{F}_{N\text{-}\mathsf{ROT}+}$ on input b' to obtain a new string, $\mathcal{S}$ returns a fresh random string $r_{b'}$. Let R' be the set sent by P_B in step 4. $\mathcal{S}$ defines B to be the set containing all values b' previously queried by P_B such that $r_{b'} \in R'$, and sends this to $\mathcal{F}_{\mathsf{SetInc}}$.

In the ideal execution, P_A outputs 1 if and only if $a \in B$. In the real execution, P_A outputs 1 if $r_a \in R$; as in the case of a corrupt P_A, this always holds if $a \in B$, and only occurs with probability $\leq |B| \cdot 2^{-s}$ otherwise. □

Efficiency. The cost of the above protocol is precisely the cost of 1-out-of-N random OT, plus sending $s \cdot |B|$ bits. If the protocol is run in a large batch using $\mathcal{F}^{\kappa,m}_{N\text{-}\mathsf{ROT}+}$ for large m (which is possible for the application to private set intersection) then this gives an amortized cost of $n_{\mathcal{C}} + s \cdot |B|$ bits per execution, where $n_{\mathcal{C}}$ is the length of the code. The costs for this when instantiated with BCH codes (as described previously) are illustrated for various choices of k in Table 2, and compared with the Walsh-Hadamard code used in [PSSZ15]. In practice, the set size used in the set inclusion subprotocol for PSI in [PSSZ15] is around $|B| = 20$. For a large item length of $k = 128$ bits, and $s = 40$-bit statistical security, this gives a 3.3x reduction in communication for the dominant component of PSI.

k	Cost with BCH (bit)	Cost with W-H (bit)
32	$467 + s \cdot \|B\|$	$1024 + s \cdot \|B\|$
64	$499 + s \cdot \|B\|$	$2048 + s \cdot \|B\|$
128	$708 + s \cdot \|B\|$	$4096 + s \cdot \|B\|$

Table 2: Comparing the communication cost of private set inclusion subprotocols on k-bit strings and size $|B|$ sets with statistical security s.

6 Implementation

We now evaluate the complexity of our random OT protocol, and compare its performance to a passively secure variant by analysing implementation results.

Complexity Analysis. The main overhead introduced by our protocol to produce m OTs, compared with the passively secure KK protocol, is the computation of $m \cdot s$ XORs (on n_C-bit strings) by each party, and the communication of $s \cdot m$ random bits from the sender to receiver, followed by $s \cdot (n_C + k_C)$ bits in the other direction, in the consistency check. However, the $s \cdot m$ bits can be reduced to κ by having $\mathrm{P_S}$ send only a single random seed for these values, and expanding the seed using a PRG.

Outside of the consistency check, the main protocol costs are the encodings, hash function evaluations and the n_C bits that are sent by $\mathrm{P_R}$ for each extended OT. Of course, the sender's computational cost also highly depends on the number of random OT outputs that are desired.

Implementation. We evaluated our protocol on two machines running over a 1Gbps local network, and also simulated a WAN environment with 50Mbps bandwidth and 100ms round-trip latency to model a real-life scenario over the Internet. All benchmarks have been run on modern Core i7 machines at 2–3 GHz.

Our implementation is in plain C, and uses the SimpleOT [CO15] oblivious transfer software [6] to run the base OTs, and BLAKE2 [ANWOW13] for hashing, as this provides fast hashing on short inputs. Otherwise, it does not rely on any other software and is available in the public domain. The executable occupies $280K$.

The core protocol covers roughly 200 lines of C code. It mostly runs on single thread, except we use OpenMP to parallelize the encodings and hash function evaluations, which are the computational bottlenecks of the protocol. We fix the computational security parameter $\kappa = 128$ and statistical security parameter $s = 40$. We used Intel AVX instructions to efficiently implement vector addition, componentwise product, and matrix transposition. Encoding of the binary linear code is implemented with multiplication by the generator matrix.

Our results for 1-out-of-N random OT for the small sized codes (repetition, Walsh-Hadamard, punctured Walsh-Hadamard and binary Golay) in the LAN setting can be seen in Fig. 6, for varying numbers of OTs. Table 3 compares the performance of the active and passively secure variants in both LAN and WAN settings, including the BCH-511 code, which could be used for the private set intersection application. We see that the overhead of active security is around 20–30% of the passive protocol over a LAN, and less than 5% in the WAN setting. This fits with the fact that the main cost of the check is computation, which is less significant in a WAN. The table also gives the amount of communication required for each choice of N, which shows that this reflects the main total cost of the protocol. Encoding of larger BCH codes (for $N = 2^{76}$) does have a noticeable effect in a LAN, though: here, BCH runtimes are 3 times higher than Walsh-Hadamard, but only have twice the communication cost. We expect that this could be improved by a more sophisticated encoding algorithm, rather than naive multiplication by the generator matrix.

[6] http://users-cs.au.dk/orlandi/simpleOT/

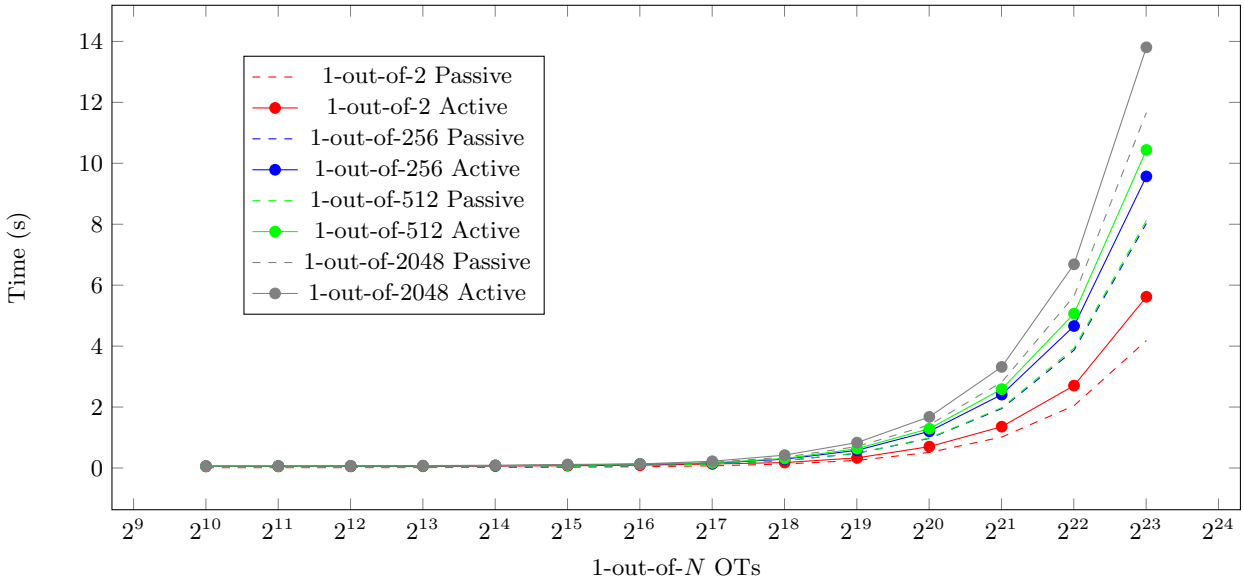

Fig. 6: Benchmarking different 1-out-of-N random OTs in LAN environment; average time for 20 runs.

Setting	$N = 2$	256	512	2048	2^{76}
Comms. (bit)	128	256	256	384	512
LAN, passive	4.1812	8.0260	8.1193	11.6642	23.4738
LAN, active	5.6191	9.5693	10.4379	13.8065	25.4001
WAN, passive	27.3982	54.2414	54.274	81.0548	108.89
WAN, active	27.882	54.7445	54.8189	81.6644	109.44

Table 3: Data transmitted per OT and runtimes in seconds for 2^{23} OTs (LAN) or 2^{20} OTs (WAN), for several choices of N

Acknowledgements We thank Ranjit Kumaresan for providing us with an extended version of [KK13].

The work in this paper has been partially supported by the ERC via Advanced Grant ERC-2010-AdG-267188-CRIPTO and the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070.

References

ALSZ13. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM Conference on Computer and Communications Security, CCS'13*, pages 535–548, 2013.

ALSZ15. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2015, Sofia, Bulgaria, April 26-30, Proceedings, Part I*, pages 673–701, 2015.

ALSZ16. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions. *Journal of Cryptology*, pages 1–54, 2016.

ANWOW13. Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-OHearn, and Christian Winnerlein. Blake2: simpler, smaller, fast as md5. In *Applied Cryptography and Network Security*, pages 119–135. Springer, 2013.

Bea96. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 479–488. ACM, 1996.

CDD+16. Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, and Jesper Buus Nielsen. Rate-1, linear time and additively homomorphic UC commitments. In *Advances in Cryptology - CRYPTO 2016, Santa Barbara, CA, USA, August 14-18, Proceedings, Part III*, pages 179–207, 2016.

CO15. Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *Progress in Cryptology - LATINCRYPT 2015, Guadalajara, Mexico, August 23-26*, pages 40–58, 2015.

EGL85. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

FJNT16. Tore Kasper Frederiksen, Thomas P. Jakobsen, Jesper Buus Nielsen, and Roberto Trifiletti. On the complexity of additively homomorphic UC commitments. In *Theory of Cryptography - TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 542–565, 2016.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology-CRYPTO 2003*, pages 145–161. Springer, 2003.

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61. ACM, 1989.

KK13. Vladimir Kolesnikov and Ranjit Kumaresan. Improved ot extension for transferring short secrets. In *Advances in Cryptology–CRYPTO 2013*, pages 54–70. Springer, 2013.

KKRT16. Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *ACM Conference on Computer and Communications Security, CCS*, 2016.

KOS15. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *Advances in Cryptology - CRYPTO Santa Barbara, CA, USA, August 16-20*, pages 724–741, 2015.

KOS16. Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *ACM Conference on Computer and Communications Security, Vienna, Austria*, pages 830–842, 2016.

Lam16. Mikkel Lambæk. Breaking and fixing private set intersection protocols. Cryptology ePrint Archive, Report 2016/665, 2016. http://eprint.iacr.org/2016/665.

LOS14. Enrique Larraia, Emmanuela Orsini, and Nigel P Smart. Dishonest majority multi-party computation for binary circuits. In *Advances in Cryptology–CRYPTO 2014*, pages 495–512. Springer, 2014.

NNOB12. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology–CRYPTO 2012*, pages 681–700. Springer, 2012.

NP99. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, Georgia, USA*, pages 245–254, 1999.

PSSZ15. Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium, Washington, D.C., USA, August 12-14, 2015.*, pages 515–530, 2015.

PSZ14. Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *23rd USENIX Security Symposium*, pages 797–812, San Diego, CA, August 2014.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008*, pages 554–571, 2008.

Rab81. Michael O Rabin. How to exchange secrets with oblivious transfer. 1981.

SSR08. Bhavani Shankar, Kannan Srinathan, and C. Pandu Rangan. Alternative protocols for generalized oblivious transfer. In *Distributed Computing and Networking, 9th International Conference, ICDCN*, 2008.

Yao82. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

Auxiliary Supporting Material

A Correlation Robust Function

We recall the standard definition of correlation robust [IKNP03] and strong correlation robust function [ALSZ15].

Let χ be a discrete probability distribution, the *min-entropy* of χ is defined as $H_\infty(\chi) = -\log(\max_x \Pr[\chi = x])$, and intuitively the min-entropy of a distribution is a measure of how predictable the distribution is.

Let $\mathcal{U}^\ell$ be the uniform distribution over strings of length ℓ.

Definition 2 (Correlation Robustness).

1. *An efficiently computable function* $\mathsf{H} : \mathbb{F}_2^\kappa \to \mathbb{F}_2^\ell$ *is* correlation robust *if*

$$(\mathbf{t}_j, \mathsf{H}(\mathbf{t}_j + \mathbf{s})) \stackrel{c}{\equiv} \mathcal{U}^{m\cdot\kappa + m\cdot\ell},$$

   where $\mathbf{t}_j, \mathbf{s} \in \mathbb{F}_2^\kappa$ *are uniformly and independently distributed.*
2. $\mathsf{H} : \mathbb{F}_2^\kappa \to \mathbb{F}_2^\ell$ *is* strongly correlation robust *if*

$$(\mathsf{H}(\mathbf{t}_j + \mathbf{s})) \stackrel{c}{\equiv} \mathcal{U}^{m\cdot\ell},$$

   where $\mathbf{s} \in \mathbb{F}_2^\kappa$ *is uniform.*

Equivalently, by defining $F_\mathbf{s}(t) = \mathsf{H}(\mathbf{t} + \mathbf{s})$, H is a correlation robust function if and only if F is a weak pseudorandom function, and H is strongly correlation robust if and only if F is a (non-adaptive) pseudorandom function.

B Instantiating the Binary Linear Code

Recall that $\mathcal{C}$ is an $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]$ binary linear code, and the protocol requires $k_\mathcal{C} = \log_2 N$ and $d_\mathcal{C} \geq \kappa$ to achieve computational security parameter κ.

Repetition Code. If $N = 2$ then the repetition code $\mathcal{C}_{\mathsf{rep}} = \{0^\kappa, 1^\kappa\}$ gives a 1-out-of-2 OT extension protocol that is very similar to the previous, most efficient protocol [KOS15] (our consistency check is slightly more expensive, but still has negligible cost for many OTs).

Walsh-Hadamard Codes. The Walsh-Hadamard code of dimension $k_\mathcal{C} = \log_2 N$ is defined by taking the inner product (over $\mathbb{F}_2$) of the message with all N length-$k_\mathcal{C}$ vectors:

$$\mathcal{C}_{\mathsf{WH}}(x) = (\langle x, y\rangle)_{y \in \{0,1\}^{k_\mathcal{C}}}$$

This has length N and minimum distance $d_\mathcal{C} = N/2$, so is an $[N, \log_2 N, N/2]$ code.

A more efficient choice is the *punctured Walsh-Hadamard code*, which restricts y to be one in the first position:

$$\mathcal{C}^*_{\mathsf{WH}}(x) = (\langle x, y\rangle)_{y \in \{1\}\times\{0,1\}^{k_\mathcal{C}-1}}$$

This has length $N/2$ and can be shown to have minimum distance $N/4$, so is an $[N/2, \log_2 N, N/4]$ code. These are slightly better parameters than the standard W-H code.

To use $\mathcal{C}^*_{\mathsf{WH}}$ for 1-out-of-N OT we simply need to ensure that $d_{\mathcal{C}} = N/4 \geq \kappa$. So, fixing $N = 512$ gives us a $[256, 9, 128]$ code, allowing 1-out-of-512 OTs at a cost of sending just 256 bits, with 128-bit security. [7]

Smaller values of N. If $N < 4\kappa$ then we can repeat the $[N/2, \log_2 N, N/4]$ punctured W-H code $4\kappa/N$ times to obtain a code of length 2κ and distance $N/4 \cdot 4\kappa/N = \kappa$, as required for security. So, with $\kappa = 128$ this has the same cost as 1-out-of-512 OT, for any power of two $N \leq 512$.

BCH codes. For larger values of N, BCH codes can be used to give much better parameters than Walsh-Hadamard codes.

For example, the $[511, 76, 171]$ BCH code would allow $N = 2^{76}$, with 171-bit security. Since in practice n will be much smaller than this, the code can be shortened to get parameters $[511 - c, 76 - c, \geq 171]$, and hence $N = 2^{76-c}$, for any desired choice of c.

Binary Golay code. The binary Golay code is a $[24, 12, 8]$ binary linear code, often used in communications. Extending this code (with repetition) 16 times gives a $[384, 12, 128]$ code, which can be used for 1-out-of-2048 OT, at the cost of sending 384 bits, with 128 bits of security. This improves upon both the Walsh-Hadamard and BCH instantiations for the same size of N.

Non Two-Power Values of n. If n is not a power of two, then we run into the problem that no binary code $\mathcal{C}$ can contain exactly n codewords. If $\mathcal{C}$ is chosen to have more than N, then the receiver may choose a value $w_i \notin \{0, \ldots, n-1\}$, which still defines a valid codeword. The consistency check will still go through, but at the end of the protocol the receiver will not obtain a valid output. However, in many applications this may not pose a problem, and the OT functionality can easily be modified to model this possibility.

[7] The punctured W-H code provides no benefit for $N > 512$, since this would cost at least 512 bits of communication, and building 1-out-of-N OT from smaller 1-out-of-N' OTs (similar to [NP99]) would be more efficient.